REMARKS

In view of the foregoing amendments and following remarks responsive to the Office Action of October 21, 2005, Applicant respectfully requests favorable reconsideration of this application.

Rejections As To Form

In Sections 3-5 of the Office Action, the Office rejected claim 1 under 35 U.S.C. § 112, second paragraph, as being indefinite. Specifically, the Office noted that claim 1 recited "said data units" without sufficient antecedent basis.

The Office's point is well taken. In fact, in reviewing the present application in view of this rejection, Applicant has noted that "said data units" also appeared in several other claims. Applicant has herein amended all such claims to now refer to "said objects" in order to correct this clerical error.

Prior Art Rejections

The Office further rejected claims 1-9 and 11-15 under 35 U.S.C. § 103(a) as obvious over applicant's admitted prior art (AAPA) in view of Kumar. The Office further rejected claim 10 under 35 U.S.C. § 103(a) as obvious over AAPA in view of Kumar and further in view of Francis.

Applicant respectfully traverses the prior art rejection. Particularly, the Office asserted that essentially all of the steps recited in claim 1 are described in Applicant's background section and particularly on page 7, lines 1-15 and in steps 104-107 of Figure 1. The Office conceded, however, that AAPA does not specifically disclose that, in the step of building the resource, the resource is built for containing the

requested object, but not containing the requested object.

However, the Office said that this was found in Kumar at column 6, lines 38-65.

The Present Invention

The present invention is an efficient method for exchanging data between two computer application programs or between a resource library and an application program. In accordance with a specific exemplary embodiment of the invention, XMI documents for transporting the data between the two applications are built on-the-fly rather than being stored in memory. Further, when a resource library or application program receives a request for an object, the resource library creates the resource to which that object corresponds, but does not populate that resource. Next, the resource library populates the resource with only the object(s) requested. Then the resource is returned to the requesting application program.

The AAPA

The above described invention is very different from the prior art described on page 7 of the present application and shown in the flow chart of Figure 1. In the flow chart shown in Figure 1, the resource factory does not create the resources on-the-fly, nor does it populate the resource with only the object requested. Rather, it stores fully populated resources in memory. It retrieves them when an object in that resource is requested and sends the entire resource to the requesting application program, even if only a single object in that resource was requested. This is wasteful and inefficient.

Discussion

The present invention solves the aforementioned inefficiency of the AAPA and other problems.

In any event, it should be apparent that the prior art described hereinabove and in the background section of the present application does not, in fact, include a resource factory "including a plurality of software modules for building resources from a data source". The resources are already built. Furthermore, it does not incorporate the steps of "responsive to a request for an object from a first computing entity, selecting a software module for building a resource of the type to which said requested object corresponds", "building a resource for containing said requested object..., said resource populated with information defining said resource, but not containing said requested object", or "inserting said requested object into said resource". These are all limitations of claim 1.

The resource factory described in the AAPA does not build the resources. It retrieves the already built resources from a data store.

Kumar has been cited for the sole alleged teaching of building the resource first without populating it with objects and then populating the resource with the objects. Thus, Kumar does not add any of the above-discussed teachings lacking from AAPA. Accordingly, independent claim 1 clearly distinguishes over AAPA.

All of claims 2-9 and 11-15 depend from claim 1 and, therefore, distinguish over AAPA for at least the same reasons.

Applicant notes that, in the Office Action dated September 23, 2004, the Office made a rejection very similar to the present rejection. Particularly, the Office rejected claims 1 and 13-15 under 35 U.S.C. § 102(b) as anticipated by AAPA. The Office withdrew that rejection in view of Applicant's arguments in response to that Office

Action filed on January 4, 2005. (The Office instead issued a new rejection based on different prior art, which Applicant also overcame).

It does not appear that the additional citation of the Kumar reference addresses any of the arguments that Applicant had used to overcome this similar previous rejection and that the Office obviously accepted. Accordingly, it is unclear why the Office is issuing a rejection that, for all intents and purposes, it has already conceded that Applicant has overcome.

In any event and assuming for the sake of argument that Kumar does teach first building a resource without populating it and then populating the resource with the objects, it still would be essentially irrelevant. Specifically, Kumar concerns an entirely different context then the present invention that is largely irrelevant to the present invention. More particularly, Kumar concerns the building of caches in object oriented programming environments. The portion of Kumar cited by the office, namely, column 6, lines 38-65, at relates to ensuring coherency between caches and the original source of data when a client 201 changes the cache object itself rather than a copy of the cached object. This portion of Kumar points out that the general paradigm for assuring this is to use the cloning process of Java for cloning the CacheEntry instances 215. CacheEntry instances 215 function to declare and maintain a cached object. Column 5, lines 19-20. A web site developer defines a CacheEntry instance 215 for each type of data that is to be cached. The CacheEntry instance is then defined to contain the hash table or vector of hash tables rather than the raw data elements.. The basic functionality includes defining methods for obtaining a cached object 215 from the cache using a cached object identifier. The CacheEntry instance 215 includes data structures holding a timestamp or other time indicator of the last

access time, a value indicating when the cache entry expires, and a cached object itself. Column 5, line 49-column 6, line 10.

However, some objects are not cloneable. In such cases, Kumar discloses:

> For objects that do not implement the cloneable inteface [sic], the clone ( ) method causes an exception to be thrown. GetCachedObject( ) preferably includes exception handling logic for creating a new object of the same type as the cached object and copying each member variable of the cached object to the new object. In essence these steps make every object cloneable for purposes of the cache in accordance with the present invention even when the object is not implement the cloneable interface.

Thus, contrary to the present invention and just like AAPA, Kumar discloses populating the CacheEntry instance 215 with all of its objects, rather than just the requested object.

The bottom line is that AAPA and Kumar, taken individually or in combination, do not disclose the concepts of the present invention as claimed of (1) building documents for transporting the data between the two applications on-the-fly rather than being stored in memory, and, (2) when a resource library or application program receives a request for an object, the resource library creates the resource to which that object corresponds, and does not fully populate that resource but, instead, populates the resource with only the object(s) requested.

Furthermore, Kumar is not even prior art to the present invention in any event. Particularly, Kumar is a U.S. patent issued November 18, 2003 and having a filing date of September 1, 2000. Accordingly, Kumar could only qualify as prior art under 35 U.S.C. 102(e) and only if Applicant did not invent the claimed invention prior to that date. In fact, Applicant did invent the invention prior to that date and has already submitted proof establishing that. Particularly, Applicant has already filed a declaration of all of the inventors under 37 C.F.R. 1.131 swearing behind the Dorsett reference

date of January 1, 2001. The Office accepted that declaration as establishing a date of invention prior to January 1, 2001. A review of that declaration and the supporting documentation, however, will show that the declaration actually also supports an invention date prior to the effective date of the Kumar reference, i.e., September 1, 2000. Particularly, although the declaration states that the dates have been redacted from the supporting documents 1-4, in fact, they were not redacted. Documents 1-3 all bear dates prior to September 1, 2000. Document 4 bears a date of September 21, 2000, which is a mere 20 days after Kumar's effective date. However, document 4 was referred to as supporting prior invention only in connection with claims 5, 6, 11, 12, 13, 14, and 15.

Accordingly, since all of the rejections rely in part upon Kumar, the rejection prima facie fails with respect to at least claims 1-4, and 7-10.

With respect to the remaining claims, claims 5, 6, and 11-15, all that is necessary is to establish that the inventors worked diligently toward reducing the invention to practice between September 1, 2000 and September 21, 2000.

In fact, although not expressly stated in the declaration (since there was no need to so state in connection with swearing behind the Dorsett reference), the declaration and documents, at a minimum, already support that the inventors conceived of the invention prior to September 1, 2000 and that they worked diligently toward reducing the invention to practice between September 1, 2000 and September 21, 2000. Specifically, documents 1-3 show that the inventors conceived of the invention at least by June 2000 and documents 1-4 collectively show that they were diligently working toward reducing it to practice between that time and September 21, 2001.

Accordingly, the previously submitted 131 declaration and supporting documents already establish that Kumar is not prior art in any event.

In view of the foregoing amendments and remarks, this application is now in condition for allowance. Applicant respectfully requests the Examiner to issue a Notice or Allowance at the earliest possible date. The Examiner is invited to contact the Applicant's undersigned counsel by telephone call in order to further the prosecution of this case in any way.

Respectfully submitted,

January 23, 2006

Theodore Naccarella
Registration No. 33,023
Synnestvedt & Lechner LLP
2600 Aramark Tower
1101 Market Street
Philadelphia, PA 19107
Telephone: (215) 923-4466
Facsimile: (215) 923-2189
Attorney for Applicant

TXN:pmf

M:\TNACCARELLA\CLIENTS\IBM\24929\PATENT OFFICE\RESPONSE TO 10-21-05 NONFINAL OFFICE ACTION.DOC